Acknowledgements

The BenchCouncil acknowledges the substantial contribution of AIoTBench proposal principal submitters: Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), and other organizations, and the proposal reviewer BenchCouncil AI committee. The BenchCouncil also acknowledges the work and contributions of the BenchCouncil AI Benchmark subcommittee member companies or institutions in developing the AIoTBench specification.

The BenchCouncil AIoTBench subcommittee would like to acknowledge the contributions made by the many members during the development of the benchmark specification. The list of significant contributors to this version includes Chunjie Luo (ICT, CAS), Xiwen He (ICT, CAS), Lei Wang (ICT, CAS), Fan Zhang (ICT, CAS), Wanling Gao (ICT, CAS), Cheng Huang (ICT, CAS), Xingwang Xiong (ICT, CAS), Jianan Chen (ICT, CAS), Hainan Ye (BAFST), Tong Wu (CNIM), Runsong Zhou (CSTC), and Jianfeng Zhan (ICT, CAS).

AIoTBench Proposal Principal Submitter: Institute of Computing Technology, Chinese Academy of Sciences

AIoTBench Proposal Reviewer: BenchCouncil AI Committee

Contact: benchcouncil@gmail.com (BenchCouncil); luochunjie@ict.ac.cn (ICT, CAS)

Trademarks

BenchCouncil, BenchCouncil Benchmark and BenchCouncil AIoTBench are trademarks of International Open Benchmark Council.
BenchCouncil Membership

(as of November 2019)

Full Members

Associate Members
<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 November 2019</td>
<td>Draft 1.0</td>
<td>Mail ballot version (proposed standard)</td>
</tr>
</tbody>
</table>
1 Introduction

Due to increasing amounts of data and compute resources, the deep learning achieves many successes in various domains. Recently, researchers and engineers make effort to apply the intelligent algorithms to the mobile or embedded devices, e.g. smart phone, self-driving cars, smart home. On one hand, the neural networks are made more light-weight to adapt the mobile or embedded devices by using simpler architecture, or by quantizing, pruning and compressing the networks. On the other hand, the mobile and embedded devices provide additional hardware acceleration using GPUs or NPUs to support the AI applications. Since AI applications on mobile and embedded devices get more and more attention, benchmarking and ranking of the AI ability of those devices becomes an urgent problem to be solved.

AIoTBench aim to provide a executable mechanism to rank the inference abilities of the mobile and embedded devices. It adopts two widely used light-weight networks which are specially designed for mobile and embedded devices. A unified metric, Valid Flops Per Second (VFPS), is proposed to evaluate the performance. VFPS reflects both the inference accuracy and the speed of the AI devices. For reference implementation, each network have been implemented in three popular frameworks, tensorflow lite, caffe2, and pytorch mobile.

2 Terminology and Background

2.1 Terminology

The terminologies used in this specification are as follows.

**AI**: Artificial Intelligence

**AIoT**: Artifical Intelligence on Things. Things refer to end devices which have limited computing and memory resources comparing to datacenter servers, e.g mobile and embedded devices.

**Benchmark**: A set of programs, datasets and metrics to evaluate the relative performance of an object.

**AIoTBench**: The benchmark released by BenchCouncil to evaluate the performance of AIoT.

**MobileNetV2**: A light-weight architecture of deep neural network.

**Bottleneck**: A modular structure used in MobileNetV2.

**ShuffleNetV2**: A light-weight architecture of deep neural network.

**Block**: A modular structure used in ShuffleNetV2

**Flops**: The number of multiply-accumulates needed by a neural network to compute an inference on a single image.

**VFPS**: Valid Flops Per Second.

2.2 Background

AI achieves many successes in various domains, e.g. image recognition, natural language processing, and speech recognition. Even for a specific domain task, there are various models proposed. Moreover, there are lots of popular deep learning frameworks on which developer can implement their AI applications.
For some equipment suppliers, their productions are designed for common purpose. For example, the mobilephone could run different AI models implemented by different frameworks. So, their productions need to support diverse workloads well. Other equipment suppliers pay attention to a single purpose. Because the workload is already determined, they could optimize their design according to the target workload. Specialization philosophy simplifies the design of the AI system, and it becomes more and more popular, especially in the embedded field. AIoTBench considers both generalization and specialization scenes through reporting the average performance and the best performance among the workloads.

3 Benchmarking Methodology

AIoTBench follows the PRDAERS benchmarking rules and methodology [1] presented by BenchCouncil. The PRDAERS is the abbreviation of paper-and-pencil, relevant, diversity, abstractions, evaluation metrics and methodology, repeatable, and scaleable.

**Paper-and-pencil Approach:** AIoTBench can be specified only algorithmically in a paper-and-pencil approach. This benchmark specification is proposed firstly and reasonably divorced from individual implementations. In general, AIoTBench defines a problem domain in a high-level language.

**Relevant:** AIoTBench is simplified and distilled of the real-world application. It abstracts the typical scenario and application of the AIoT.

**Diversity and Representatives:** Modern workloads show significant diversity in workload behavior with no single silverbullet application to optimize for [2]. Consequently, diverse workloads should be included to exhibit the range of behavior of the target applications. AIoTBench covers the typical light-weight models and development frameworks used in mobile and embedded environments.

**Abstractions:** AIoTBench abstracts the image classification task. AIoTBench is light-weight thus portable across different mobile and embedded devices.

**Evaluation Metrics and Methodology:** The performance number of AIoTBench is simple to count and understand. It reflects both the inference accuracy and the speed of the AI devices.

**Repeatable, Reliable, and Reproducible:** The testing results of AIoTBench are repeatable, reliable, and reproducible.

**Scaleable:** Models used in AIoTBench have different scale configuration to cover different parameter size and computing complexity.

4 Design

4.1 Benchmark procedure

A complete AIoTBench testing system, as shown in Fig [1], contains multiple components: data set, workload select module, inference module, and score module. The data set is indexed by ID, and placed at the permanent storage of the device in advance. The data set should be the original format, and can not be compressed or preprocessed. Workload select module allows users to choose which workload to run. Inference module executes the workload. It contains four steps, data reading, preprocessing, predicting, and logging. The time of data reading, preprocessing and predicting is counted in final executing time, since reading and preprocessing also matter the performance of AI system significantly. Logging format should be “image_id, workload_id, real_label, predict_label, time”. After all testing finished, score module analyzes the log file, and computes the final score.
4.2 Task

Currently, AIoTBench contains the task of image classification. A classifier network takes an image as input and predicts its class. Image classification is a key task of pattern recognition and artificial Intelligence. It is intensively studied by the academic community, and widely used in commercial applications of AIoT. Image classification is also widely used in other AI benchmarks, and becomes a de facto standard to evaluating AI system. ImageNet classification dataset [3] is used, which has 1280000 training images and 50,000 validation images with 1000 classes. In AIoTBench, all classification models are trained on the ImageNet 2012 training set. And the inference uses the ImageNet 2012 validation set.

4.3 Model

AIoTBench adopts two widely used light-weight networks as the reference model. These models are specifically tailored for mobile and embedded devices with constrained resources.

4.3.1 MobileNetV2

The key module of MobileNetV2 [4] is the inverted residual with linear bottleneck. It expands a low dimensional compressed representation to a high dimension by a 1x1 pointwise convolution, then the high dimensional features are filtered with a lightweight 3x3 depthwise convolution. Finally, features are projected back to a low dimensional representation with a linear 1x1 pointwise convolution. This module can be easily implemented in any modern framework. Also, it allows a particularly memory efficient implementation.

AIoTBench uses two configurations of MobileNetV2: tiny and standard. The detail configuration of MobileNetV2 can be seen in [1]. For tiny configuration (mobilenet_v2_0.35_192), there are 1.66M parameters, and 43M flops complexities. It achieves 58.2% accuracy on ImageNet 2012 validation data. For standard configuration (mobilenet_v2_1.0_224), there are 3.47M parameters, and 300M flops complexities. It achieves 71.8% accuracy on ImageNet 2012 validation data.

4.3.2 ShuffleNetV2

ShuffleNetV2 [5] is designed by some guidelines and empirical studies, e.g. 1) use equal channel width convolutions; 2) avoid using group convolution; 3) reduce the degree of fragmentation; and 4) reduce element-wise operations like ReLU and shortcut connection. To maintain a large number and equally wide channels with neither dense convolution nor too many groups, the input of feature channels are split into two branches. One branch remains as identity, and the other branch consists of 1x1 pointwise, 3x3 depthwise, 1x1 pointwise convolution, with the same input and output channels. After convolution, the two branches are concatenated and shuffled.
Table 1: MobileNetV2, for tiny and standard configuration of complexities.

AIoTBench also uses two configurations of ShuffleNetV2: tiny and standard. The detail configuration of ShuffleNetV2 can be seen in [2]. For tiny configuration (shufflenet_v2_0.5x), there are 1.41M parameters, and 41M flops complexities. It achieves 60.3% accuracy on ImageNet 2012 validation data. For standard configuration (shufflenet_v2_1.5x), there are 3.5M parameters, and 299M flops complexities. It achieves 72.6% accuracy on ImageNet 2012 validation data.

Table 2: ShuffleNetV2, for tiny and standard configuration of complexities.

4.4 Metrics

It is a consensus that benchmarking of AI system should consider the accuracy. The AIoT system should compromise between speed and accuracy. There are two ways to consider the accuracy in existing AI benchmarking. 1) MLperf Inference Benchmark [6] requires that all implementations achieve a quality target within 1% of the reference model’s accuracy. It is unfair to such system that achieves a accuracy within 2% of the reference models accuracy, but speedup 10x or 100x performance. 2) AI Benchmark [7] considers the accuracy as a score feature among lots of other features. The weight of each feature is hard to determine since the features are incomparable with each other.
AloTBench uses a unified metric, Valid Flops Per Second (VFPS), to evaluate the performance. VFPS reflects both the inference accuracy and the speed of the AI devices. Flops (some literatures use MACs) is the number of multiply-accumulates needed to compute an inference on a single image. It is a common metric to measure the computation complexity of the model. AloTBench uses the Valid Flops Per Second (VFPS) as the metric to evaluate the inference performance of AI devices. VFPS is defined as following:

\[ VFPS = \frac{\sum I_i(predict\_\text{label}, real\_\text{label}) \times F_i}{\sum T_i} \]  

(1)

Where \( F_i \) refers to the flops of the model used in the inference. \( T_i \) refers to the time. \( I \) is an indicator function:

\[ I = \begin{cases} 
1 & \text{if predict\_\text{label} equals to real\_\text{label}} \\
\alpha & \text{else} 
\end{cases} \]  

(2)

Where \( \alpha \) is a penalty factor which is set to \(-0.1\). VFPS counts the computations which predict correctly. Meanwhile, it punishes the incorrect prediction by the penalty factor. When testing, all workloads are executed respectively. And VFPS is computed for each workload. We report two final scores: average VFPS and maximum VFPS.

4.5 Benchmarking Constraints

AloTBench should satisfy the following constraints.
1) AloTBench allows the adjusting of setting of software and hardware on which the the workloads are running.
2) AloTBench disallows modification of the workloads, including retraining of the model, model pruning, and reducing the input data size.

4.6 Measuring Procedure

AloTBench result submission should contain the following information: the hardware and software configurations, VFPS score of each workload, the average VFPS score, and the maximum VFPS score. All the data are uploaded to BenchHub, a code management system used to host source code and manage projects. The BenchHub’s web address is www.benchcouncil.org/benchhub. After the submission, AloTBench community will check the validity, repeatability, and authenticity of the results. If the results are abideance by the rules, the submission will be reported online.

5 Reference Implementation

5.1 Framework

For the mobile and embedded devices, the framework, with which the models are implemented, is also part of the workload. In AloTBench, each model is implemted by three frameworks: Tensorflow lite, Caffe2, Pytorch mobile.

5.1.1 Tensorflow Lite

TensorFlow, released by Google, is a free and open-source software library for dataflow and differentiable programming. It is widely used for machine learning applications such as neural networks. Tensorflow uses static computational graphs. Tensorflow lite is released for deploying the models trained by tensorflow on mobile and embedded devices. After the model is trained, it need be converted to a .pb graph, and then executed on mobile or embedded devices using the tensorflow lite interpreter, available on Android as well as iOS platforms.
5.1.2 Caffe2

Caffe is an open-source deep learning framework, originally developed at UC Berkeley. Caffe2, built on the original Caffe and released by Facebook, is a light-weight and modular framework for production-ready training and deployment. Its mobile version supports iOS and Android platforms. Caffe2 uses static computational graphs and nodes representing various operators. The deploy of the model on mobile and embedded devices is similar with tensorflow lite.

5.1.3 Pytorch Mobile

Pytorch, primarily developed by Facebook’s AI Research lab (FAIR), is an open-source machine learning library based on the Torch library. It aims to replace for NumPy to use the power of GPUs and provide a deep learning research platform. PyTorch uses dynamic computational graphs. PyTorch mobile supports an end-to-end workflow from Python to deployment on iOS and Android. The deploy of the model on mobile and embedded devices is similar with tensorflow lite.

5.2 Workloads

There are 12 workloads in AIoTBench. The details of the workloads are listed in Table 3:

<table>
<thead>
<tr>
<th>id</th>
<th>model</th>
<th>framework</th>
<th>network scale</th>
<th>flops</th>
<th>parameters</th>
<th>accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1</td>
<td>MobileNetV2</td>
<td>tensorflow lite</td>
<td>tiny</td>
<td>43M</td>
<td>1.66M</td>
<td>58.2</td>
</tr>
<tr>
<td>W2</td>
<td>MobileNetV2</td>
<td>pytorch mobile</td>
<td>tiny</td>
<td>43M</td>
<td>1.66M</td>
<td>58.2</td>
</tr>
<tr>
<td>W3</td>
<td>MobileNetV2</td>
<td>caffe2</td>
<td>tiny</td>
<td>43M</td>
<td>1.66M</td>
<td>58.2</td>
</tr>
<tr>
<td>W4</td>
<td>MobileNetV2</td>
<td>tensorflow lite</td>
<td>standard</td>
<td>300M</td>
<td>3.47M</td>
<td>71.8</td>
</tr>
<tr>
<td>W5</td>
<td>MobileNetV2</td>
<td>pytorch mobile</td>
<td>standard</td>
<td>300M</td>
<td>3.47M</td>
<td>71.8</td>
</tr>
<tr>
<td>W6</td>
<td>MobileNetV2</td>
<td>caffe2</td>
<td>standard</td>
<td>300M</td>
<td>3.47M</td>
<td>71.8</td>
</tr>
<tr>
<td>W7</td>
<td>ShuffleNetV2</td>
<td>tensorflow lite</td>
<td>tiny</td>
<td>41M</td>
<td>1.4M</td>
<td>60.3</td>
</tr>
<tr>
<td>W8</td>
<td>ShuffleNetV2</td>
<td>pytorch mobile</td>
<td>tiny</td>
<td>41M</td>
<td>1.4M</td>
<td>60.3</td>
</tr>
<tr>
<td>W9</td>
<td>ShuffleNetV2</td>
<td>caffe2</td>
<td>tiny</td>
<td>41M</td>
<td>1.4M</td>
<td>60.3</td>
</tr>
<tr>
<td>W10</td>
<td>ShuffleNetV2</td>
<td>tensorflow lite</td>
<td>standard</td>
<td>299M</td>
<td>3.5M</td>
<td>72.6</td>
</tr>
<tr>
<td>W11</td>
<td>ShuffleNetV2</td>
<td>pytorch mobile</td>
<td>standard</td>
<td>299M</td>
<td>3.5M</td>
<td>72.6</td>
</tr>
<tr>
<td>W12</td>
<td>ShuffleNetV2</td>
<td>caffe2</td>
<td>standard</td>
<td>299M</td>
<td>3.5M</td>
<td>72.6</td>
</tr>
</tbody>
</table>

Table 3: The workloads of AIoTBench

References


